VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2011

Jan Deutschl

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 6. květen 2011 .....

Rád bych na tomto místě poděkoval spolupracovníkům z firmy XEVOS Solutions za jejich pomoc a podmětné rady.

Abstrakt

Obsahem této práce je popis úkolů a jejich řešení, které byly zadány v rámci odborné praxe u firmy XEVOS Solutions. Zadané úkoly se týkají vývoje internetových aplikací na platformě ASP.NET. Zejména v oblasti CMS a ERP systémů.

Klíčová slova: XEVOS Solutions, ASP.NET, CMS,ERP

Abstract

The content of this thesis is the description of tasks and their solutions which have been solved at the professional practice at the XEVOS Solutions company. Assigned tasks are related to the development of Internet applications using ASP.NET. In particular the CMS and ERP systems.

Keywords: XEVOS Solutions, ASP.NET, CMS,ERP

Seznam použitých zkratek a symbolů

ERP	–	Enterprise Resource Planning
CMS	–	Content Managment System
AJAX	–	Asynchronous JavaScript and XML
XML	–	Extensible Markup Language
API	–	Application Programming Interface
HTML	–	Hyper Text Markup Language
CSS	–	Cascade Style Sheet
DLL	–	Dynamic-link library
LINQ	–	Language Integrated Query
MS	–	Microsoft
ASCII	–	American Standard Code for Information Interchange
CSV	–	Comma Separated Value
ORM	–	Object-relational mapping
SEO	–	Search Engine Optimalization

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

1.1 Profil firmy a pracovní zařazení studenta

XEVOS Solutions poskytuje od svého založení v roce 2006 komplexní služby v oblasti informačních a komunikačních technologií s důrazem na inovativnost a vysokou dostupnost realizovaných řešení. Společnost se zaměřuje především na konzultace, realizace a správu serverů, síťové infrastruktury a klientských stanic v segmentu firem působících na území celé republiky. V roce 2008 se specializace firmy rozšířila o vývoj webových aplikací na platformě ASP.NET, kde mezi pilotní projekty patří vývoj ERP systému postaveného čistě jako webová aplikace. Tato expanze vyústila v roce 2008 založením společnosti XEVOS Solutions s.r.o. a posílením týmu o nové členy. Dnes se orientuje také na Business Inteligence a inovuje stávající aplikace o funkce využívající BI. [1]

Ve firmě XEVOS Solutions jsem pracoval na pozici ASP.NET developer, podílel jsem se na vývoji ERP a CMS aplikací, které firma vlastní a na informačním systému pro společnost Skyzzo.

1.2 Seznam úkolů s vyjádřením časové náročnosti

Odhad časových náročností úkolů, které byly zadány v rámci praxe a kterým se tato práce dále věnuje. Uvedené časy jsou včetně analýzy, implementace, testování a případné opravy chyb u zadaných úkolů. Nezahrnují však čas nutný ke studiu použitých technologií a čas, který byl nutný k zorientování se v architektuře projektu, na kterém se v rámci úkolu pracovalo.

- Galerie fotografií
  - Celkový čas 49h
- Vytvoření serverového ovládacího prvku pro ořez fotografie
  - Celkový čas 32h
- Implementace autentizace pomocí služby MojeID
  - Celkový čas 40h
- Filtrace dat v ovládacím prvku RadGrid
  - Celkový čas 16h
- Filtrování dat nad agendou transakcí
  - Celkový čas 30h
- Agenda výplat
  - Celkový čas 80h

- Agenda CSV souborů
  - Celkový čas 32h
- Analýza výkonu ERP systému
  - Celkový čas 16h
- Optimalizace databázové vrstvy
  - Celkový čas 18h
- Snížení objemu dat odesílaných servrem na klienta
  - Celkový čas 35h
- Virtual Scrolling
  - Celkový čas 32h

1.3 Používané technologie a nástroje

- MS Visual Studio 2010 Ultimate – integrované prostředí pro návrh, správu a vývoj aplikací postavených na technologii od společnosti Microsoft.
- MS Managment Studio 2008 R2 – Nástroj pro správu databáze MS SQL Server.
- Team Foundation Server 2010 – platforma pro spolupráci týmu v rámci životního cyklu aplikace.
- MS SQL Server 2008 - Relační databáze společnosti Microsoft.
- C# 4.0 - Vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý firmou Microsoft.
- ASP.NET - Framework pracující nad knihovnou .NET a sloužící k tvorbě dynamických internetových aplikací a webových služeb.
- ASP.NET AJAX Framework– Rozšíření od společnosti Microsoft. Obsahuje množství nových serverových ovládacích prvků využívajících AJAX. Připravuje javascriptové API na straně klienta, které pak využívají jiné serverové ovládací prvky a webové služby. Umožňuje asynchronní částečné obnovení stránky pomocí ovládacího prvku UpdatePanel.
- Telerik RadControls for ASP.NET – sada serverových ovládacích prvků společnosti Telerik, které jsou založeny na ASP.NET AJAX. Jedním z nejvíce propracovaných prvků je RadGrid sloužící k zobrazení dat formou tabulky. Všechny prvky mají podrobně popsány své API, které lze využít na straně serveru i klienta na webových stránkách společnosti Telerik [2]

2 Zadané úkoly a jejich řešení

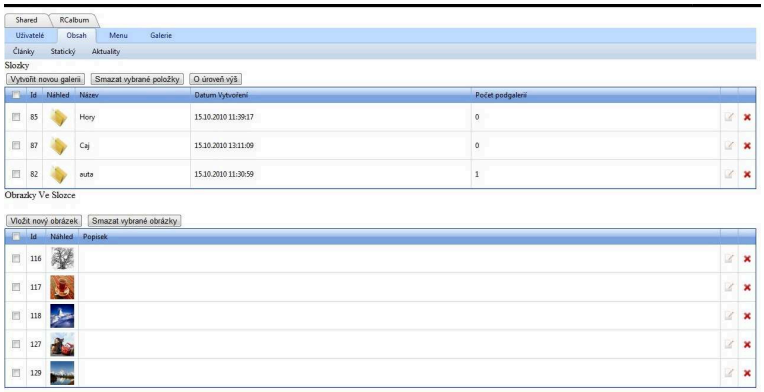
2.1 XEVOS CMS Live

XEVOS CMS Live je nástroj pro správu obsahu menších až středně velkých webových prezentací. Celý systém je postaven jako webová aplikace na platformě ASP.NET 3.5. Obsahuje moduly pro publikování článků s využitím pokročilého WYSIWYG editoru, správu statického textu na webu a fotogalerie. Jedná se o zákaznické řešení, kdy se aplikace přizpůsobuje potřebám konkrétního zákazníka. Při tvorbě webu je kladen důraz na SEO vytvářených stránek, uživatel systému může editovat hlavičku každé stránky včetně obsahu meta informací. Systém automaticky používá pretty url, tedy přepis url adres stránek na straně serveru tak, aby formát adresy byl pro uživatele přívětivější a čitelnější.

2.1.1 Galerie obrázků

2.1.1.1 Cíl Vytvoření modulu pro nově vznikající verzi systému CMS firmy Xevos pro práci s obrázky. Galerie se skládá z alb a obrázků. Každé album může obsahovat libovolný počet obrázků a dalších alb, čímž vznikne stromová struktura galerie s neomezeným vnořováním. Uživatel může obrázky a alba editovat, mazat a měnit pořadí, ve kterém se budou zobrazovat návštěvníkům stránek. Obrázky se do galerie nahrávají hromadně, pokud rozměry obrázku přesáhnou určitou velikost, je uživateli nabídnuto buď vybrat z obrázku určitou oblast, která se vyřízne a uloží nebo změnit rozměry obrázku automaticky. Pokud jsou obrázky ve správném formátu, uloží se do databáze, kde je lze případně dále editovat.

2.1.1.2 Postup Hlavní část agendy tvoří dva ovládací prvky RadGrid. Nad každým z nich jsou ovládací tlačítka pro smazání a vložení nové položky. První RadGrid zobrazuje jednotlivá alba. Pomocí funkcionality 'Drag & Drop', která je v komponentě RadGrid podporována, je možné měnit pořadí řádků. Kliknutím na řádek je uživatel přesměrován do konkrétního alba. K posunu nahoru slouží tlačítko 'O úroveň výš'. Pokud uživatel smaže nějaké album, rekurzivně se odstraní všechna alba a fotografie, které mazané album obsahovalo. Druhý prvek RadGrid slouží k zobrazení fotografií v albu. Fotografie je možné nahrávat a mazat pomocí tlačítek v záhlaví.



Obrázek 1: Ukázka galerie obrázků

2.1.2 Serverový ovládací prvek pro ořez fotografie

2.1.2.1 Cíl Vytvořit serverový ovládací prvek pro zmenšení velikosti fotografie vybráním oblasti, která se má uložit s použitím knihovny JCrop[5]. Tento prvek bude v systému CMS použitý na základě požadavku zákazníka, aby bylo možné ukládat fotografie jen v určitém poměru stran. Pokud tedy fotografie nahraná uživatelem nebude odpovídat potřebnému poměru stran bude mu umožněno vybrat oblast, která se uloží a použije.

2.1.2.2 Postup Knihovna JCrop zajistí grafické znázornění výběru oblasti obrázku. Pracuje pouze na straně klienta. Pokud uživatel změní vybranou oblast, uloží se její souřadnice do skrytého pole na stránce. Až se stránka odešle zpět na server jsou souřadnice přečteny a obrázek je na serveru zpracován.

Serverové ovládací prvky jsou třídy v jazyce C#, které dědí z třídy System.Web.UI.Control nebo z odvozených tříd. Tyto třídy přímo do stránky generují HTML kód. ASP.NET nabízí velké množství serverových ovládacích prvků, které jsou již připraveny k použití jako například Button (tlačítko), TextBox (vstupní pole), případně komplikovanější například GridView (tabulkový pohled), který lze navázat i na zdroj dat. Ovládací prvek pro ořez fotografií dědí přímo ze třídy Systém.Web.UI.WebControls.Image, protože jeho základem, stejně jako této třídy, je HTML tag *img*. Pokud se prvek přidá na stránku, je nutné zaregistrovat knihovnu JCrop a css soubor, který tato knihovna používá. Klientský skript se registruje do stránky před renderováním obsahu (tedy html elementu *img*) v události OnPreRender. Z důvodu lepší čitelnosti a přehlednosti stránky se vkládá do elementu head. Toho se docílí zavoláním metody RegisterClientScriptBlock objektu ClientScript. Protože je ovládací prvek součástí jiného projektu (assambly), než webová stránka, nelze k souborům v tomto projektu přistupovat z webové stránky přímo. Aby se do stránky vložily potřebné soubory scriptů a stylů je nutné na ně přistupovat pomocí webového zdroje. Webové zdroje používají ovladač WebResource.axd. Tento ovladač příjmá požadavky na url, extrahuje zdoj z patřičné assambly a poté vrací obsah. Url adresa zdroje má dva parametry. Parametr *d* který odkazuje na zdroj do assambly a *t*, který značí časové razítko. Přes tento ovladač je možné přistupovat k souborům, které jsou součástí externí dll knihovny z webové stránky. Tyto soubory musí mít nastavenou vlastnost Build Action na Embedded Resource.

Obrázek 2: Ukázka ovládacího prvku pro ořez fotografie

```
<script src="/WebResource.axd?d=MhKym-HMXZ7ZZpmLOfTJy6DE584DHWHxKdpr8j0GXq-
    ZyyaoqMMppm4lo2ejxRvxt20Xmb3qlj5TE-WTQzNxDCf4ytc_M5D2BltSDl483VHQOljE62g9g-
    dEN2v2XgmeHWvnICcc_dcuBq9sBN886yxOiZySqoRzB4fo74iVCdq4v2Et5WdJ5KmUqLfzM17A0
    &amp;t=634371088661026948" type="text/javascript"></script>
```

Výpis 1: Příklad použití webového zdroje

2.1.3 Implementace autentizace pomocí služby MojeID

2.1.3.1 Cíl Cílem tedy je vytvořit ukázkovou implementaci autentizace pomocí služby mojeID a navázat ji na současný systém ověření uživatele v produktu CMS Live.

2.1.3.2 Postup MojeID nabízí uživatelům internetu centralizované přihlašování na všechny weby, které tuto službu podporují. Provozovatelem je sdružení CZ.NIC. Uživatel si tedy na webových stránkách www.mojeid.cz založí svůj profil, kde vyplní údaje. Pod jednotným uživatelským jménem a heslem se pak může přihlašovat k více webům, navíc je pouze v jeho rukách, jaké údaje chce webové stránce na kterou se chce přihlásit poskytnout. Minimum je samozřejmě uživatelské jméno. Výhodou služby MojeID je, že pokud stránka vyžaduje např. adresu nebo telefonní číslo uživatele a uživatel svolí k poskytnutí těchto údajů je zaručeno, že tyto údaje jsou validní. Registrovaní uživatelé služby MojeID totiž procházejí několikastupňovou validací údajů. Předpokládá se, že autentizace pomocí služby MojeID bude jen jednou z možností ověření identity uživatele a bude zachován stávající systém autentizace. Celý proces přihlašování je založen na knihovně openID, kterou využívá například i portál seznam.cz. Získání identity probíhá ve dvou fázích:

1. Pomocí knihovny OpenID vytvoříme požadavek (instanci objektu IAuthenticionRequest), do kterého vložíme přihlašovací jméno uživatele se sufixem .mojeid.cz. K požadavku dále připojíme výčet povinných atributů, které potřebujeme po uživateli, aby nám poskytl. Požadavek odešleme, čímž uživatele přesměrujeme na stránky mojeid.cz. Zde uživatel vyplní své heslo a potvrdí všechny atributy, které chce stránce zveřejnit. Nakonec uživatel akci potvrdí, čímž se přesměruje zpět na naši stránku.

2. Z odpovědi serveru MojeID získáme objekt typu IAuthenticionResponse. Nyní musíme ověřit, jestli autentizace proběhla úspěšně. Rozhraní IAuthenticionResponse obsahuje vlastnost status. Pokud je nastavena na Authenticated znamená to, že můžeme uživatele považovat za autentizovaného. V tomto kroku tedy ještě uložíme údaje o přihlášení do naší databáze a předáme řízení dalších kroků stávajícímu autentizačnímu systému, který se postará o kroky jako vytvoření autenizační cookie atd. V ostatních případech stavu vlastnosti status oznámíme uživateli, že proces autentizace selhal.

2.2 Informační systém Skynet

Skynet je multilevel systém pro pražskou společnost. Účelem je evidence osob, které pro společnost pracují, jejich uzavřených pojistných smluv a interních platebních transakcí.

2.2.1 Filtrace dat v ovládacím prvku RadGrid

2.2.1.1 Cíl Pro zobrazení dat v tabulce byl použit bohatě vybavený serverový ovládací prvek RadGrid. Tento prvek mimo jiné obsahuje funkcionalitu vyhledávání ve sloupcích tabulky, která je realizována dvěma prvky. Vstupním polem a tlačítkem. Oba jsou umístěny v záhlaví tabulky. Do vstupního pole (textboxu) se napíše řetězec, který se má vyhledat a tlačítkem, které vyvolá menu se vybere filtr, který se má na záznamy aplikovat. Filtr může nabývat hodnot jako obsahuje, rovná se, je větší než atd.

Zjistilo se, že uživatelé informačního systému nejčastěji filtrují data podle jednoho typu porovnání a to obsahuje. A tak by bylo pro ně pohodlnější napsat do políčka pro vyhledávání pouze řetězec, který chtějí vyhledat a zmáčknout klávesu enter, čímž by se vyhledávání ihned spustilo. Tuto funkcionalitu však ovládací prvek RadGrid přímo nepodporuje.

2.2.1.2 Postup Při tvorbě ovládacího prvku RadGrid se naváže na vstupní pole (textbox) vyhledávání událost klientská onkeydown, která zajistí volání následující metody javascriptu při každém stisknutí klávesy na vstupním poli.

```
function filterOnEnterContains(sender, eventArgs) {
    if (eventArgs.keyCode == 13) {
        eventArgs.cancelBubble = true;
        eventArgs.returnValue = false;
        if (eventArgs.stopPropagation) {
            eventArgs.stopPropagation();
            eventArgs.preventDefault();
        }
        var masterTableView = \$find("<%= RadGrid1.ClientID %>").
            get_masterTableView();
        var index = sender.parentNode.cellIndex; //index of the current column
        var columns = masterTableView.get_columns();
        var uniqueName = columns[index].get_uniqueName();
        masterTableView.filter(uniqueName, sender.value, Telerik.Web.UI.
            GridFilterFunction.Contains);
    }
}
```

Výpis 2: Metoda pro spuštění vyhledávání

Tato metoda využívá API, které poskytuje prvek RadGrid na straně klienta. Pokud má stisknutá klávesa kód 13, což je ASCII kód pro klávesu enter je nejdříve zabráněno defaultnímu chování filtrace. Poté je získána reference na komponentu RadGrid a název sloupce, který má být filtrován. Následně je zavolána metoda komponenty, která odešle požadavek a aplikuje filtraci nad daným sloupcem.

2.2.2 Filtrování dat nad agendou transakcí

2.2.2.1 Cíl Administrátorovi systému je umožněn pohled na všechny transakce, které v systému probíhají. Každá transakce si kromě jiných údajů uchovává odkaz na tabulku s typy jednotlivých transakcí. V systému jich nyní je 29. Předpokládá se ale, že se tento počet může změnit. Každá transakce patří do jedné ze čtyř kategorií, které specifikují zdali se jedná o odchozí nebo příchozí a na jaký ze dvou účtů každého uživatele transakce směřuje. Cílem tedy je umožnit administrátorovi vybrat z množiny všech typů transakcí pouze ty, o které má zájem. Tato funkcionalita je realizována tlačítkem v záhlaví tabulky, které vyvolá vyskakovací okno se seznamem všech transakcí v systému. Vedle každého typu je zaškrkávací tlačítko, které daný typ přidá k výběru. Po zvolení požadovaných transakcí se volba potvrdí tlačítkem použít.

2.2.2.2 Postup Základem stránky s typy transakcí je ovládací prvek checkBoxList. Tento prvek jde svázat se zdrojem dat z relační databáze. Jetliže se tedy přidá nový záznam do tabulky transakčních typů, je přidáno i nové zaškrkávací tlačítko v okně. Dále jsou v záhlaví stránky další čtyři zaškrkávací tlačítka, které reprezentují kategorie transakcí. Pokud uživatel některé toto tlačítko zaškrkne, automaticky se mu označí i transakce, které do kategorie spadají. Toto je realizováno pomocí javascriptu, aby nedocházelo ke zbytečnému odesílání stránky zpět na server. Ve chvíli, kdy uživatel dokončí výběr a potvrdí formulář, zavolá se metoda na stránce, která vyskakovací okno otevřela a předají se jí v parametru vybrané kategorie oddělené středníkem. Poté se vyskakovací okno zavře. Stránka, která obsahuje seznam transakcí se odešle na server, kde se na základě vybraných typů transakcí vyfiltrují data a výsledek se odešle klientovi.

2.2.3 Agenda výplat

2.2.3.1 Cíl Vytvořit samostatnou agendu, která získává data o hodnotě účtů uživatelů z databáze a prezentuje je ve formě tabulky. Umožnit procházení historie výplat s možností zvolit časový interval. Umožnit spuštění generování výplat pro určitou množinu uživatelů. Graficky znázornit data, kdy došlo ke generování výplat.

2.2.3.2 Postup Nejdříve bylo potřeba vytvořit získat data pro agendu. Nad tabulkou uživatelů a transakcí byl vytvořen pohled, který obsahovuje identifikační údaje uživatelů a k nim hodnoty stavu účtu, které již byly vyplaceny, a které na vyplacení čekají. Tento pohled slouží jako zdroj dat pro serverový ovládací prvek RadGrid, který zobrazuje tato data ve formě tabulky. Pokud uživatel v daném měsíci ještě nebyl vyplacen, označí se záznam červeně, pokud již ano tak zeleně. Do agendy má přístup pouze administrátor. Nad tabulkou jsou umístěny dva kalendáře, kterými jde volit časový úsek, který se má zobrazit a jsou v nich zvýrazněna data, kdy došlo ke generování výplat. Při výběru data z kteréhokoliv kalendáře je zajištěno, že se v druhém kalendáři automaticky nastaví datum tak, aby obě vybrané data byla ve stejném měsíci a roce. Administrátor má možnost označit určitý počet záznamů a tlačítkem generovat výplaty spustí proceduru vyplacení výplaty. V tuto chvíli se také vloží záznam do databáze o generování, aby bylo možno procházet historii vyplácení výplat.

2.2.4 Agenda CSV souborů

2.2.4.1 Cíl Administrátor může vložit do systému externí csv soubor obsahující data o plnění pojistných smluv uživatelů systému. Cílem je vytvořit agendu umožňující správu a ukládání těchto souborů.

2.2.4.2 Postup Agenda obsahuje tabulku, ve které administrátor vidí seznam všech CSV souborů, které byly do systému nahrány s informacemi o zdroji dat, datumu vytvoření a zpracování souboru a počtu záznamu v daném souboru. Dále je mu poskytnut odkaz ke stažení souboru. Agenda obsahuje formulář k vložení nového souboru, kde se vyplní zdrojová banka, nebo pojišťovna, která soubor dodala a odkaz na soubor. Po odeslání formuláře se soubor uloží na disk serveru. Protože se soubor ukládá ve složce App_Data, která je systémová a klient k souboru tedy nemůže přistoupit přímo pomocí url adresy zadané do prohlížeče je pro stažení souboru využit handler. Handler je speciální soubor s příponou .ashx, který obsahuje jednu metodu s názvem ProcessRequest a jednu vlasnost s názvem IsReusable, která značí jestli se jedna instance handleru dá využít pro více požadavků. Při spuštění handleru se zavolá metoda. K handleru lze přistupovat pomocí url adresy podobně jako ke stránce. Parametrem metody ProcessRequest je objekt HttpContext, který obsahuje vlasnost Response tedy odpověď. Pomocí knihoven System.IO se vytvoří ze souboru pole bytů a vložit je do objektu Response pomocí metody write. Pokud tedy uživatel v tabulce s CSV soubory klikne na odkaz ke stažení odešle požadavek na tento handler a vrátí se mu soubor csv.

2.3 XEVOS ERP Live

Informační systém je určen pro evidenci a správu dat menších firem, které se zabývají nákupem a prodeje. Základní vlastnosti systému je evidence:

- Zboží - tedy informace o jeho ceně, místě uskladnění, dostupnosti a dalších vlastnostech
- Faktur - s informacemi o způsobu platby, dodavateli případně odběrateli v závislosti na tom zda se jedné o fakturu vydanou nebo přijatou. Evidencí položek, jejich množství a ceny na dané faktuře a dalších údajů.
- Objednávek - zde se rozlišují přijaté a vydané. Evidují se podobné informace jako u faktury.
- Dodacích listů - požadavek evidence naskladnění nebo vyskladnění.
- Skladů a firem - je umožněno evidovat dva typy skladů - fyzický a virtuální.
- Osob - tedy uživatelů systému, kteří využívají jeho funkcí.

Data uchovává v relační databázi MS SQL Server. Systém je postaven jako webová aplikace na platformě ASP.NET.

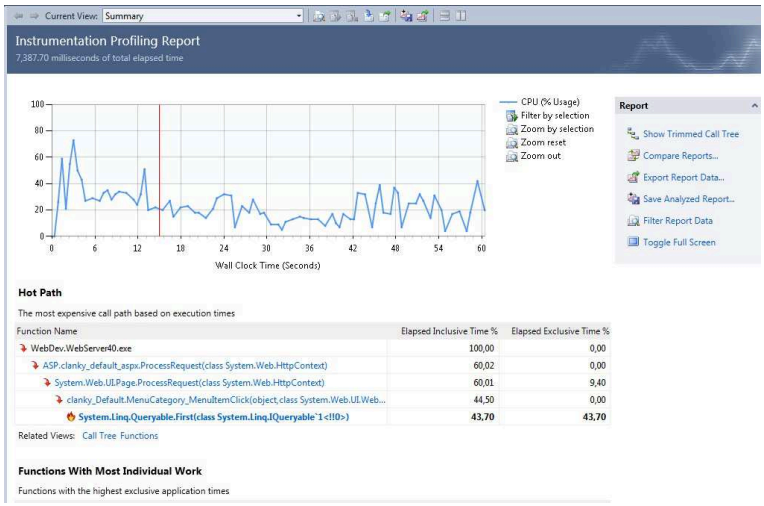
2.3.1 Analýza výkonu systému

2.3.1.1 Cíl Pomocí nástrojů Visual Studia 2010 zanalyzovat výkon systému ERP Live. Navrhnout způsoby optimalizace rychlosti systému.

2.3.1.2 Postup Pro analýzu výkonu systému používá Visual Studio nástroj zvaný profiler, který je k dispozici pouze ve verzi Ultimate. Lze jej spustit ve třech módech:

- CPU Sampling - Vyhodnocuje jak aplikace zatěžuje procesor
- Instrumentation - Zaznamenává kolikrát se která funkce volá a jaký čas se v ní stráví.
- .NET Memory Alocation - Vyhodnocuje alokovanou paměť pro aplikaci.

Po spuštění profileru se spustí i aplikace a během práce s ní se sbírají analytická data. Z těchto dat se pak vytváří reporty. Po otevření reportu je k dispozici strom volaných metod a tzv. hot path. Hot path ukazuje na metodu, ve které program během profilování strávil nejvíce času. V detailu jednotlivých volání lze pak zjistit další analytická data včetně výpisu kódu metody. Z profilování systému vyšly najevo některé nedostatky databázové vrstvy aplikace, které způsobovaly zbytečné zpomalování systému.



Obrázek 3: Ukázka z výstupu profileru Visual Studia 2010

2.3.2 Optimalizace databázové vrstvy

2.3.2.1 Cíl Na základě výsledků z analýzy výkonu systému zoptimalizovat metody pro přístup k datům z relační databáze.

2.3.2.2 Postup Pomocí profileru se zjistilo, že při přístupu k datům se nejvíce času stráví v metodě GetOrdinal objektu SqlDataReader. Každá třída objektově relačního modelu databáze má metodu, která vrací instanci této třídy naplněnou daty odpovídajícím jednomu řádku tabulky. Tato metoda jako má jako parametr objekt, který implementuje rozhraní System.Data.IDataRecord. Ten umožňuje na základě indexu vrátit hodnotu atributu řádku tabulky. Místo indexu je ale použita metoda GetOrdinal, která na základě názvu atributu tento index vrací. Zavolání metody GetOrdinal vyžaduje přístup k systémovým tabulkám databáze, a pokud je volána v cyklu může ovlivnit výkon aplikace. Pokud je vyžadováno více řádků z tabulky, volá se metoda pro vrácení instance objektu ORM v cyklu, což způsoboje volání v cyklu i metody GetOrdinal.

Před započetím cyklu se vytvoří slovníková kolekce, která obsahuje název sloupce a index všech atributů tabulky. V cyklu se pak místo metody GetOrdinal volá tato kolekce. Tuto změnu bylo nutné provést pro všechny třídy ORM.

2.3.3 Snížení objemu dat odesílaných serverem na klienta

2.3.3.1 Cíl Dalším krokem v optimalizaci a zrychlení systému je snížit čas, po který se stránka posílá ze serveru na klienta. Toto se týká hlavně agendy faktur, kdy naměřený čas příjmání stránky s fakturou, která má pětset položek trval 4,5 sekundy.

2.3.3.2 Postup Některé části aplikace odesílaly na klientský počítač až 2MB dat při každém požadavku. To bylo mimo jiné způsobeno velikostí ViewState. ViewState je skryté pole, které ASP.NET automaticky vkládá do každé stránky. Slouží k uložení dynamického nastavení ovládacích prvků na stránce mezi jednotlivými požadavky. Pokud je těchto prvků na stránce větší množství, může velikost ViewState dosáhnout stovek KB dat. Byly implementovány dvě metody jak zmenšit velikost skrytého pole ViewState. Prvni je komprimace pomocí Gzip. ASP.NET obsah pole šifruje, ale nekomprimuje. V životním cyklu stránky[3] se volají dvě metody, které pracují s ViewState.

- LoadPageStateToPersistenceMedium() - Pokud se nejedná o první požadavek, provede se tato metoda ihned po inicializaci stránky. Dekóduje a uloží stav do objektu ViewState.
- SavePageStateToPersistenceMedium(object state) - Provede se před renderováním stránky. Uloží obsah ViewState do skrytého pole na stránce.

Při ukládání se objekt obsahující ViewState nejdříve serializoval a převedl na pole bajtů, poté zašifroval a zkomprinoval s použitím algoritmu GZip. Výsledek se pak uložil do skrytého pole na stránce. Při načítání byl postup pouze opačný. Komprimací se velikost ViewState zmenší až o 40%.

V některých případech, kdy i tak velikost ViewState nebyla optimální, se použil jiný přístup. Obsah ViewState se uložil do objektu Cache na serveru a klientovi se ve skrytém poli odeslal pouze jednoznačný identifikátor uložení v cache. Tato možnost však zbytečně zabírala paměť'serveru, protože se Cache v nejhorším případě uvolní až při vypršení doby session.

Pro další úsporu přenášených dat se ještě minimalizoval veškerý skript a soubory stylů pomocí nastroje Ajax Minifier[4].Ten odstraní všechny prázdné znaky, zalomení řádků a názvy vnitřních proměnných ve funkcích změní na jednoznakové.

2.3.4 Virtual Scrolling

2.3.4.1 Cíl V detailu záložky faktury se zobrazí výpis všech položek, které jsou svázány z danou fakturou. Jejich počet se může pohybovat od několika desítek do stovek. Protože uživatele ne vždy zajímá tento seznam celý, je cílem tedy navrhnout způsob, jak ušetřit množství dat přenesených na klienta omezením velikosti tohoto seznamu.

2.3.4.2 Postup Požadavkem je, aby se data ke klientovi stahovala až ve chvíli, kdy o ně požádá. Inspirací byl systém načítání obsahu, který je znám například z internetového prohlížeče Google při prohlížení obrázků. Google nezobrazí hned všechny obrázky, ale pokud uživatel dojde až dolů k poslednímu automaticky se vyvolá požadavek a vrátí se další. Tomuto systému se říká Virtual Scrolling. Virtual scrolling je způsob postupného načítání obsahu. Obsah musí byt rozdělen na pevně dané úseky, např. 20 záznamů v tabulce. Klientovi se s prvním požadavkem odešle pouze první strana a teprve až bude chtít přejít na další vznikne požadavek, který ze serveru získá další data a aktualzuje tabulku. Nevýhodou je časová prodleva mezi jednotlivými požadavky na další strany způsobená čekáním na odpověď serveru. Výhodou je pak, že se klientovi odešlou pouze data, o které opravdu stojí a ušetří se tak množství přeneseného obsahu.

Virtual Scrolling byl v tomto případě použit na seznam položek v kartě faktury. Seznam byl realizován serverovým ovládacím prvkem RadGrid společnosti Telerik, který už má pro Virtual Scrolling zabudovanou podporu a umožňuje jej použít ve dvou módech:

1. Pokud se klient dostane k poslednímu záznamu, je zavolána služba, která vrátí další množinu záznamů, vloží je na konec tabulky a aktualizuje vlastnost komponenty, která udržuje aktuální počet záznamů. Celkový počet záznamů tedy není znám, dokud uživatel nepřejde na poslední stránku. Tento přístup je vhodný použít v případě, když očekáváme, že klient bude chtít data číst sekvenčně. Výhodou je, že již jednou načtená data se neztrácí a pokud se klient chce vrátit o několik záznamů zpět, je to již bez prodlevy.

2. V tomto případě je předem známý celkový počet záznamů v tabulce a v prohlížeči je aktuálně načtena vždy jen jedna stránka se záznamy. Uživatel se může v tabulce pohybovat pomocí rolování. Pokud chce přejít na další stránku vytvoří se událost needDataSource a zašle se požadavek na server. V obslužné metodě události se získají z databázového serveru data pro další stránku, naplní se jimi tabulka a odešle se odpověď. Předchozí stránka není v prohlížeči uchována.

V tomto úkolu byl použit druhý přístup, protože se předpokládá, že uživatel nebude k položkám na faktuře přistupovat pouze sekvenčně. Pro zajištění rychlejší odezvy se pro aktualizaci gridu použil serverový ovládací prvek UpdatePanel, který zajistí, že se požadavek odešle asynchronně s využitím AJAXu.

S vyžitím komprimace popsané v předchozím úkolu a Virtual Scrolling se podařilo zmenšit objem dat odesílaných klientovi v agendě faktur z původních 2,1 MB na 0,5 MB. Čas potřebný k získání stránky se zmenšil z průměrných 3,5 s na 1,3 s.

3 Závěr

3.1 Znalosti získané během studia uplatněné v průběhu praxe

Během individuální praxe jsem využil znalosti získané během studia, které se týkaly prací s relačními databázemi a jazykem SQL. Využíval jsem znalostí jazyka T-SQL při psaní procedur a triggerů v databázi. Dále používal technologii LINQ k přístupu k datům a jazyk C#, se kterými jsem také měl zkušenosti. Během práce na systému ERP Live jsem využíval znalostí složitostí a optimalizace algoritmů.

3.2 Znalosti scházející v průběhu praxe

Ve firmě se používal systém pro řízení a správu životního cyklu aplikace, založený na Team Foundation Serveru, který usnadňoval týmovou práci nad jedním projektem a zajišťoval větvení a správu verzí vyvíjené aplikace. S tímto systémem jsem neměl žádné zkušenosti. Dále mi chyběla znalost databázového projektu visual studia, který zajišťuje správu schématu databáze, jeho verzování a další funkce.

3.3 Celkové hodnocení

Získal jsem mnoho cenných zkušeností z oblasti vývoje internetových aplikací a práce v týmu. Zdokonalil jsem znalosti vývoje založeném na platformě ASP.NET. Během praxe jsem se zúčastnil školení na téma testování aplikací a vývoje na platformě Windows Azure, které firma zajistila.

4 Reference

[1] *XEVOS Solutions* [Online] Dostupné z:
http://www.xevos.cz/?stranka=20-firemni-profil

[2] *Telerik ASP.NET AJAX Controls* [Online] Dostupné z:
http://www.telerik.com/products/aspnet-ajax.aspx

[3] *ASP.NET Page Life Cycle Overview*[Online] Dostupné z:
http://msdn.microsoft.com/en-us/library/ms178472.aspx

[4] *Microsoft Ajax Minifier 4.0*[Online] Dostupné z:
http://www.asp.net/ajaxlibrary/AjaxMinDocumentation.ashx

[5] *Knihovna JCrop*[Online] Dostupné z:
http://deepliquid.com/content/Jcrop.html

[6] Matthew MacDonald, Mario Szpuszta
ASP.NET 3.5 a C# 2008 tvorba dynamických stránek profesionálně, Zoner Press, 2008

[7] Mickey Gousset, Brian Keller, Ajoy Krishnamorthy, Martin Woodward
Řízení životního cyklu aplikací ve Visual Studiu 2010, Zoner Press, 2010.